Ph.D. Dissertation Proposal
MGLAIR: A Multimodal Cognitive Agent Architecture

Jonathan Bona
Department of Computer Science and Engineering
201 Bell Hall
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000
jp bona@buffalo.edu

Committee Chair: Dr. Stuart C. Shapiro
Committee Members: Dr. William J. Rapaport, Prof. Josephine Anstey

June 30, 2010
Abstract

I propose to develop and implement a multimodal, grounded, layered, cognitive architecture for embodied agents. The field of cognitive architectures aims to replicate the types of cognitive functionality present in the human mind. The ability to sense and act simultaneously in different modalities is a key feature of humans and other biological cognitive agents. The proposed architecture advances the state of the art through its general treatment of modalities. This treatment includes a conception of a modality as a configurable unitary resource with its own properties that determine its role within an agent’s sensing and acting capabilities, and mechanisms within the architecture to integrate multimodal sensing and acting with reasoning. The architecture will make possible the creation of more capable and more cognitively plausible agents than can be made using existing architectures without these features. This effort will include developing the theoretical foundations of the architecture; an implementation in software of the architecture as a platform for the development of computational cognitive agents; and a new implementation of the SNePS acting system, which will be an integral part of the overall architecture.

1 Introduction

I propose to develop a computational theory and implementation of a multimodal, grounded, layered, cognitive architecture for embodied agents. The field of cognitive architectures is generally concerned with the specification and development of systems that are capable of producing the types of cognitive functionality present in the human mind. These include human-level reasoning capabilities, perception, memory systems, and acting capabilities. Often, such architectures are used as starting points for the implementation in software of cognitive agents that carry out reasoning, perceiving, remembering, acting, etc., according to the architecture’s models of those phenomena.

The proposed architecture will serve as a platform for the implementation of cognitively plausible embodied computational agents capable of reasoning and planning. As embodied agents situated in some environment, these must be capable of perceiving, interpreting, representing, and acting within the world.

The creation of computational cognitive agents may be motivated primarily by their usefulness as tools to perform specific tasks, or by a need to test models and theories of cognition by implementing and using them. These motivations are not necessarily exclusive - agents designed to test cognitive theories may well prove to be actually useful as well. The goal for this project is a general-purpose architecture that is suitable for both practical and theoretical applications. This architecture will provide the underpinnings for embodied cognition without restricting agents’ capabilities to an overly-specific model of the human brain or of human cognition. Significant features of the architecture include a layered structure, a logic-based reasoning system, an emphasis on support for embodied and situated agents, an acting system that connects reasoning with planning to act and with acting in the world, and a model of multimodal perception and action. These are discussed in more detail in sections 3 and 4.

The architecture is layered in that it is organized into three main components that can be viewed as arranged in a vertical stack. From the top down, these are: a knowledge layer (KL), a perceptuo-motor layer (PML), and a sensori-actuator layer (SAL) [Hexmoor et al., 1993b].

At the top, the KL and its subsystems form the locus of conscious reasoning. All of an agent’s mind or mental capabilities are implemented in this layer. The KL is centered around a logic-based representation and reasoning system with syntax and semantics similar to standard first order logics.

Representation and reasoning are tied to an acting system within the KL that contains agents’ goals and plans. These are connected to the agent’s beliefs about the world, and the KL contains mechanisms to carry out those plans by acting.

At the bottom, the SAL includes low-level controls for sensors, motor capabilities, and other effectors. Its implementation is specific to the embodiment and environment in which the agent operates.

Between the KL and the SAL is the PML, which connects the SAL to the KL via three successively more abstract sublayers.

The main focus of this proposal is the multimodal nature of the architecture. Humans and other naturally occurring, i.e., biological, cognitive agents make constant use of the ability to sense and act in multiple
modalities simultaneously.

By “modality”, I mean both the basic afferent sensory abilities that combine to make up an agent’s perceptual experiences (e.g., vision, hearing, etc) and the efferent capabilities that allow it to act and affect the world (e.g., speech and a multitude of motor capabilities). Each modality operates independently of the others as a limited resource through which a particular type of act or sense data is managed.

Support for concurrent multimodal sensing and acting is a central feature of the proposed architecture. This feature will make possible the creation of computational cognitive agents that are more cognitively plausible than, and that are more capable than, agents implemented on architectures that do not support concurrent sensing and acting in multiple modalities.

Both sensing and acting involve the flow of information between layers of the architecture. The architecture divides this flow into independent data channels for each of the available modalities. This is achieved by establishing the notion of a modality as a type of object with its own data channel and properties such as direction and priority.

The role of modalities in this architecture is described and discussed in more detail in section 6. Determining the exact model and implementation of modalities, and creating mechanisms to integrate reasoning with multimodal perception and action, constitute the bulk of the proposed work.

2 Defining Cognitive Architecture

Alan Newell coined the term *cognitive architecture* in *Unified Theories of Cognition* [Newell, 1994], in which he was concerned specifically with human cognition and with *Soar* as a model of human cognition:

> Our ultimate goal is a unified theory of human cognition. This will be expressed, I have maintained, as a theory of the architecture of human cognition — that is, of the fixed (or slowly varying) structure that forms the framework for the immediate processes of cognitive performance and learning. [Newell, 1994, p. 111]

There are at least as many definitions of *cognitive architecture* as there are instances of cognitive architectures. What many of these have in common is the notion of a specification of a system in terms of its parts, their functionality, and the integration of and interactions between distinct parts, such that the whole is sufficient for cognition, or “mind”. Section 2.1 presents several such definitions by progenitors of cognitive architectures. It is followed by my own working definition in section 2.2.

2.1 Definitions

John Anderson, creator of ACT-R [Anderson, 1996] defines *cognitive architecture* as:

> ...a specification of the structure of the brain at a level of abstraction that explains how it achieves the function of the mind [Anderson, 2007, p. 7].

This definition reflects Anderson’s (and ACT-R’s) approach to cognition, which is focused on creating models that resemble the human brain, with modules that correspond to functional units in the brain and that process information in a similar manner.

John E. Laird, co-creator of Soar (along with Paul Rosenbloom and Alan Newell) defines a *cognitive architecture* as:

> ...the fixed infrastructure that supports the acquisition and use of knowledge. ... A cognitive architecture consists of:

- memories for storing knowledge
- processing units that extract, select, combine, and store knowledge
- languages for representing the knowledge that is stored and processed [Laird, 2008, p. 2]
Ron Sun, creator of the hybrid neural network-based architecture, CLARION, defines it as follows:

A cognitive architecture provides a concrete framework for more detailed modeling of cognitive phenomena, through specifying essential structures, divisions of modules, relations between modules, and a variety of other aspects [Sun, 2004, p. 341].

[Hexmoor et al., 1993a] in an early paper on GLAIR as an embodied agent architecture characterize cognitive architecture as a type of architecture “for understanding/modeling behaviors of an anthropomorphic agent”, noting that cognitive architectures are concerned with “the relationships that exist among the structures of memory, reasoning abilities, intelligent behavior, and mental states and experiences” and often ignore the body to focus narrowly on modeling the mind.

While this last point remains true of some work in the field, it is now more common for mainstream cognitive architectures to be used as platforms to implement embodied cognitive agents capable of acting and interacting in the (real or simulated) world [Best and Lebiere, 2006] [Rickel and Johnson, 2000].

2.2 Working Definition of Cognitive Architecture

My working definition of cognitive architecture combines the notions of embodiment and agency with the idea of a framework for cognition exemplified by the other definitions above. In this proposal and related work, I use cognitive architecture in the following sense:

A cognitive architecture is a high-level specification of a system in terms of its parts, their functionality, and the interactions between those parts such that the system may be used as a platform on which models of cognition may be implemented and tested.

In particular, I am concerned with the type of cognitive architecture in which models take the form of cognitive agents capable not only of intelligent “thought”, but also of embodied intelligent action in a world. This view of cognitive architecture is consistent with the goals of SNePS:

The long term goal of the SNePS Research Group is to understand the nature of intelligent cognitive processes by developing and experimenting with computational cognitive agents that are able to use and understand natural language, reason, act, and solve problems in a wide variety of domains [The SNePS Research Group, 2010].

3 MGLAIR among Architectures

3.1 Introduction

The subject of this proposal is the development of a theory and implementation of a multimodal (M), grounded (G), layered (L), architecture (A) with integrated (I) reasoning (R): MGLAIR.

Reasoning is central to the architecture’s design. Its implementation of agents’ minds at the knowledge layer will be comprised mainly of a logic-based knowledge representation and reasoning system, discussed more in section 4.2. MGLAIR’s reasoning system is integrated with the rest of the architecture through its ties to the acting system, which allows agents to sense and act in the world. The architecture is layered in that it is divided into a set of interconnected modules that can be viewed as vertically stacked layers with direct connections between adjacent layers. These layers combine to create a gradation of abstractions from raw sense data in the lowest level (the body) to symbolic representations of perceived entities at the knowledge layer (the agent’s conscious mind). These symbolic mental representations in the KL are grounded in perception of the world by their alignment with perceptual structures in lower levels of the architecture. MGLAIR is multimodal in the sense that agents built using the architecture can sense and act concurrently in multiple modalities. That is, they can process and reason about sensory information coming from different, and different kinds of, sensors at once. They can also perform complex acts or multiple acts that use different, and different kinds of, effectors simultaneously. This is achieved within the architecture by establishing
modalities that connect the conscious mental agent (knowledge layer) to its physical body (sensori-actuator layer). In MGLAIR, acts, plans, and the state of the world and of things in it are represented symbolically at the knowledge layer. These symbols are ultimately grounded by alignment with structures in lower layers of the architecture. Those intermediate structures may form a sort of representation of the world, but I do not take them to be part of an explicitly multimodal representation scheme.

### 3.2 Comparison with other architectures

Most cognitive architectures have in common some elements not explicitly captured by the general working definition I provided in the previous section.

The following is a list of cognitive architectures compiled from listings in [Langley et al., 2009], [Samsonovich et al., 2010], and [Pew and Mavor, 1998]: 3T, 4D/RCS, ACT-R, AIS, APEX, ART, BECCA, biSoar, CERA-CRANIUM, CIRCA, Chrest, Clarion, CogAff, COGNET, CogPrime, CoJACK, Emile, Entropy Reduction Engine, Epic, FORR, GLAIR, GMU-BICA, HOS, HTM, Icarus, Leabra, LIDA, MIDAS, NARS, OMAR, Pogamut, Polyscheme, Prodigy, PRS, Remote Agent Architecture, RCS, SAL, SAMPLE, Soar, Tosca, and Ymir. The dissertation here proposed will contain a full literature review that examines many of these in detail.

This section gives an overview of several prominent cognitive architectures, and discusses the degrees to which they support the sort of multimodal action and perception that will be central to MGLAIR. The architectures listed below are all similar to MGLAIR in that each of them supports symbolic reasoning, mostly as enhanced versions of traditional production systems, some augmented by neural networks or other statistical elements as part of the architecture. For example, many prominent architectures (e.g., ACT-R [Anderson, 1996] and Soar [Laird et al., 1987]) take the symbolic approach to representing (mental states) and reasoning (mental processes) — though some (e.g., CLARION [Sun, 2007]) are based on connectionist models or even on hybrid models that combine the two approaches.

MGLAIR’s knowledge layer includes logical representation and reasoning capabilities and a variety of accompanying inference mechanisms that are arguably more complex than those provided by most architectures. These capabilities facilitate representations of metaknowledge and metareasoning, nested beliefs (beliefs about beliefs, etc), complex belief revision, and situatedness via alignment with the perceptuo-motor layer. This combination of features appears to be unique to MGLAIR.

#### 3.2.1 Architectures Overview

The *ACT-R* (Adaptive Control of Thought—Rational) architecture [Anderson, 1996] is based on a traditional production system, with condition-action pairs at the center of its reasoning capabilities. Modularity is a key feature of ACT-R. It divides processing among distinct modules (perceptual, motor, declarative memory, etc) operating in parallel, each of which is connected to the central, procedural, production system via buffers. Each of these modules is supposed to correspond to a distinct region of the human brain. ACT-R supports subsymbolic learning, as well as the adoption of new productions as the result of a production compilation learning mechanism [Anderson et al., 2004]. ACT-R can be considered to handle multimodal processing, in the sense that many of the built-in information processing modules connected to the central production system correspond to commonly-used perceptual and efferent modalities for embodied agents (e.g. *manual, visual, aural, vocal*) [Anderson et al., 2007]. However, unlike in MGLAIR, these processing modules are fixed in number and in nature, and altering them would require modifications to the architecture itself.

*Soar* [Laird et al., 1987] has its roots in traditional, symbolic production systems but has been extended to include modules for semantic and episodic learning and memory, a reinforcement learning module, and more. Recent extensions to Soar also include short-term and long-term visual memory to store and manipulate visual percepts [Laird, 2008]. Though Soar does not seem to address the sort of generalized multimodal perception and action I am proposing for MGLAIR, and recent work to extend Soar has focused on unrelated topics, work to “incorporate other modalities” is listed as a potential area for future research in [Derbinsky and Laird, 2010].
The bi-Soar architecture [Kurup and Chandrasekaran, 2007] is based on Soar and extends its (symbolic, production-based) representation capabilities by adding diagrammatic representations of visual data. This allows for bi-modal cognitive states and problem states, and perception and action that make use of both symbolic and diagrammatic representations. bi-Soar does not include a model for multimodal perception and action. Its notion of modalities does not include support for multiple modalities that extends beyond the bi-modal integration of symbols and diagrams.

CERA (Conscious and Emotional Reasoning Architecture) [Arrabales et al., 2009] is a layered architecture organized into the CORE, instantiation, and physical layers, which roughly correspond to the KL, PML, and SAL in MGLAIR. Its sensory processing model is based on preprocessing to generate atomic “single” percepts based on raw sensor data from a single modality at the lowest level, and on the composition of single percepts into more complex percepts. Percepts from a single modality can be combined to form multimodal percepts. For instance, sonar and visual percepts may be combined to form a single percept corresponding to an object of interest in an agent’s field of vision. As with other architectures in this list, the selection of available modalities within CERA is seems to be fixed. Descriptions of the project focus on the perceptual integration of three sensory modalities in particular for the purposes of spatial reasoning: sonar, contact, and visual.

EPIC (Executive-Process/Interactive Control) [Kieras and Meyer, 1997] is an architecture designed specifically for modeling multimodal embodied tasks. Like ACT-R, it consists of a central, production-system-based “cognitive processor” surrounded by modules for short- and long-term memory and specialized processing, including a fixed limited set of sensory and motor module processors (auditory, visual, tactile, ocular motor, vocal motor, and manual motor) for handling perceptuo-motor tasks. Each of these processes data appropriate to its modality and interfaces with the working-memory module of the cognitive processor. Unlike MGLAIR, EPIC ties its multimodal capabilities to dedicated sensory and motor processor modules and does not seem to include a general model of modality that allows the addition or alteration of modalities without modifying the architecture.

4 Background, Motivations, and Architectural Properties

4.1 Primary Motivation

Different cognitive architectures take inspiration from different fields of study. These commonly include biology, psychology, and philosophy. This project and related previous work by members the SNePS Research Group are motivated most directly by computational philosophy. That is, a major common goal of our projects has been to gain understanding of human-level intelligence without necessarily creating systems that mimic the structure or organization of the human brain [Shapiro, 1992]. Psychology and biology have also served as inspiration at times. There seems to be no clear consensus on what features are required in a system in order for it to count as, e.g., biologically inspired.

4.2 GLAIR and SNePS

MGLAIR has its roots in the GLAIR (Grounded Layered Architecture with Integrated Reasoning) cognitive agent architecture [Hexmoor et al., 1993b].

GLAIR has as a major part of its knowledge layer (the subsystems of which combine to produce conscious reasoning) the SNePS knowledge representation, reasoning, and acting system [Shapiro and The SNePS Implementation Group, 2007].

Work on GLAIR began when Kumar and Shapiro extended SNePS’s pure reasoning abilities with a subsystem capable of integrating reasoning with acting and planning [Kumar et al., 1988] [Kumar and Shapiro, 1991]. This acting system is a key part of GLAIR. In section 6.3 I describe it in more detail and propose some modifications to the next version of the acting system, which will be part of MGLAIR.

GLAIR divides agent design into a series of layers: the Knowledge Layer (KL), the Perceptuo-Motor Layer (PML), and the Sensori-Actuator Layer (SAL). The PML is further divided into distinct sublayers.
(PMLa, PMLb, PMLc), as shown in Figure 1. With the exception of those related to modality, the features of MGLAIR introduced in section 1 are generally also present in GLAIR.

At the KL, a running instance of SNePS can be viewed as the mind of a computational cognitive agent. Asserted propositions within an agent’s KL can be viewed as its first-person beliefs. Reasoning is achieved through various types of inference on the agent’s beliefs. These include natural deduction and other standard methods of logical inference, as well as methods that manipulate the underlying structure of the representation.

Memory is primarily long-term in nature, though mechanisms internal to the inference engine provide a form of short-term working memory. The short-term/long-term distinction often emphasized in other cognitive architectures is not a part of our current theory or implementation. Long-term memory within the system includes semantic memory that, for embodied agents, may be grounded by alignment with perceptually-derived structures in lower layers of the architecture, and episodic memory supported by deictic representation of the present and past.

The idea of a multimodal architecture to replace GLAIR grew out of the use of agents with some basic multimodal capabilities in intermedia virtual drama performances [Shapiro et al., 2005a]. These and subsequent agents have used distinct data channels for separate types of act and sense data. However, they lack a shared theoretical framework, a standard implementation, and a rigorous approach to the integration of multimodal perception and action with reasoning. The project proposed here improves upon GLAIR by, among other things, specifically addressing and supporting multimodal perception and action.

4.3 Cognitive Architecture Properties and Evaluation Criteria

[Langley et al., 2009] presents properties, capabilities, and criteria for evaluation of cognitive architectures. This section discusses (M)GLAIR in terms of some of the elements in this framework.
• **Representation of Knowledge:** The knowledge layer is implemented in the SNePS knowledge representation, reasoning, and acting system. Implementations of the other layers (especially the SAL, which is specific to an agent’s embodiment) vary from one agent to the next.

• **Mixture of Representational Formalisms:** SNePS itself supports multiple formalisms because of its triune nature as simultaneously logic-based, frame-based, and network-based.

• **Support for Meta-knowledge** - SNePS directly supports meta-knowledge and meta-cognition, including straightforward representation of propositions about propositions due to its use of term-based logic. This obviates the use of work-arounds such as a separate holds predicate, which is commonly used to allow propositions about propositions in first-order logics. By including in the knowledge layer a term that refers to the agent itself, many SNePS agents are able to represent and reason about themselves and perceive their own actions as sensory information reaching the KL through the PML.

• **Declarative and Procedural Representations:** The acting system contains procedural representations in the form of policies, actions and acts, as well as declarative propositions about acts. Whether it is considered as logic-based, frame-based, or network-based, the representation used for other types of knowledge in the KL is unarguably declarative.

• **Structured/Hierarchical Organization of Knowledge:** SNePS uses an inherently structured organization of knowledge. Its term-based predicate logic representation allows for “nested” proposition-valued terms that refer to other terms, including other terms that denote propositions. SNePS has often been used to represent hierarchical information, including subclass/superclass hierarchies, parthood and other mereological relations, and similar information used in ontological reasoning.

• **Generality:** GLAIR is a very general-purpose architecture. It has been used to implement agents that operate in a wide variety of environments and domains and perform diverse tasks. For example, agents (some with rudimentary multimodal processing capabilities) have been created to: act in dramatic performances (these include robotic agents acting in the real world, and simulated agents controlling avatars in virtual environments) [Shapiro et al., 2005b]; model early human mathematical cognition [Goldfain, 2008]; reason about threats in the cyber security domain [Kandefer et al., 2007]; and many more.

• **Versatility:** An advantage of GLAIR’s multi-layered architecture is that it allows agents’ minds to be moved between similar but distinct embodiments, environments, and contexts without major alterations. It is possible to develop and test a GLAIR agent in a simulated environment before moving the agent to a hardware robot in the real world (or in some cases to another simulation). The architecture itself places no restriction on the number or type of perceptual or efferent modalities an agent can use, and modalities can be defined and added with relative ease when the environment or the agent’s embodiment require it.

### 4.4 Properties Motivating Research in MGLAIR

Other properties discussed in [Langley et al., 2009] suggest areas for research on MGLAIR. In this section I discuss the relevance of several of these areas to the work I am proposing.

• **Integration of Percepts:** Perception is crucial for MGLAIR as an architecture for embodied cognitive agents. One significant aspect of perception is the integration of percepts from different modalities into a coherent experience. Existing GLAIR agents, even those with some basic division of sense data into distinct modalities, make no serious attempt to combine information from multiple modalities. Though a full treatment of this type of integration might be the subject of its own proposal, it must be addressed by MGLAIR in order to be taken seriously as a multi-modal architecture. My research and development of MGLAIR will provide a model of, and mechanisms for, perceptual integration.
• **Allocation of Perceptual Resources:** Part of the task of integrating perception will involve deciding how to allocate perceptual resources, a behavior that [Langley et al., 2009] characterizes as attention. The agent Patofil (see section 5.1) has this capability at a basic level, prioritizing reception of different percepts based on context. This project will explore how this can be implemented in a general way as part of the architecture available to all MGLAIR agents.

I will also be concerned with the allocation resources (e.g., motors) to efferent modalities during the execution of complex multiple acts that may be competing for them.

• **Execution and Action: Learning from Behavior:** Though the SNePS acting system is already well developed, I propose in section 6 some extensions and modifications that will support the implementation of MGLAIR. [Langley et al., 2009] suggests that a cognitive architecture should support learning of abilities and policies. This capability may allow agents to adapt to changes in the environment, learn new abilities (either from other agents or perhaps even through action and reflection), etc. Part of the current proposal is a new model of modality as a type of unitary resource, instances of which agents may have explicit knowledge about, and may even be able to modify themselves as they learn about their world and embodiment.

• **Interaction and Communication:** Communication between agents, including between humans and computational cognitive agents can be a crucial source of information to agents. One goal of the SNePS Research Group is the creation of natural language competent agents [Shapiro and Rapaport, 1991]. SNePS includes a parser and grammar(s) for natural language understanding of English text, and work on expanding natural language understanding within SNePS is ongoing. Perception and action are vital to natural language understanding and generation, especially in verbal communication.

We have experimented with agents that use speech generation and speech recognition software to communicate “out loud”, with limited success, in one of the virtual drama applications discussed in section 5.3. These minimally speech-competent agents used an extremely simple model of speech action and speech perception. Their utterances are represented at the KL and throughout the PML by simple symbols corresponding to English sentences or fragments. The translation to and from units of sound takes place only in the SAL. This is not a cognitively plausible model of speech generation and perception. The proposed architecture will provide direct support for better models of the use, representation, and perception of natural language, including the structure of spoken language.

5 **Current State of MGLAIR**

This section describes existing GLAIR agents with some multimodal capabilities, and tools I have created to assist with agent development. These illustrate some of the problems with the current state of multimodal GLAIR. Among these problems are the lack of a shared framework that specifies the nature, behavior, and general use of modalities; the absence of a coherent approach to the integration of multiple modalities with reasoning; no standard implementation of the full architecture; and technical issues such as the current difficulty of creating agents with multimodal capabilities. These are discussed more in section 6.1.

5.1 **Patofil Agent**

*The Trial The Trail* [Shapiro et al., 2005b] is a virtual drama that takes place in a simulated 3D world and uses GLAIR agents with some multimodal capabilities as characters in the performance. This early use of “MGLAIR” agents has been the basis for several more recent agent projects. Here I examine the specific structure of the agents used in this performance. I will take as an example the agent called Patofil.

Patofil is the main agent-actor in *The Trial The Trail*, and the agent with which the user (a human participant in the immersive 3D environment) has the most interaction. Patofil’s task is to guide the participant through a journey. The participant’s actions are largely unconstrained, so part of the agent’s task involves keeping the user engaged and on the right track. Patofil is able to “see” the participant’s
location and current actions. The capabilities of her vision modality do not include visual representation structures or processing techniques necessary for true machine vision. Rather, the environment provides the implementation of the agent’s virtual body with descriptions of those events that the agent is capable of “seeing” based on her, and the users’ and other agents’, current locations and activities. These descriptions are converted to propositional knowledge as they pass up through the layers of the architecture and into the knowledge layer.

Patofil is able to move, adjust her mood, and address the user by gesturing or speaking out loud by selecting and playing audio files of prerecorded utterances.

Patofil’s Knowledge Layer is implemented in SNePS using the SNePSLOG logical language interface. The KL includes acts, plans, beliefs, rules, and other mental content as well as frame definitions for primitive actions, and the code that associates those primitive actions to functions in the PML.

Patofil’s PMLa is implemented in Lisp and includes definitions of those primitive action functions accessible to the KL, functions to parse sensory information traveling up to the KL, separate queues for each modality, and a sense-act-loop to manage percept queues. KL terms denoting primitive actions are directly associated with (“attached to”) their implementations in the PMLa using the attach-primaction function.

Patofil’s PMLb is also implemented in Lisp. It includes functions to initialize and manage TCP/IP sockets for each modality and functions to read and write those sockets. The sockets connect the PMLb to the PMLc, and provide a way of passing acts and sense data between the two.

Patofil’s PMLc is implemented within the 3D virtual world framework Ygdrasil [Pape et al., 2003] (as is the SAL), and runs on a different physical computer from the ones used by the KL and upper layers of the PML. The primary function of the PMLc is to pass to the SAL efferent messages arriving on sockets from the PMLb, and to pass percept messages up from the SAL to the PMLb.

The SAL is implemented entirely within Ygdrasil. It includes functions in Ygdrasil’s scripting language to move the agent’s body in the world, sense relevant changes to the world, etc. Those functions are accessible to the PMLc. Figure 2 shows the implementation of the architecture’s layers for Patofil.

![Figure 2: MGLAIR for Patofil in The Trial The Trail](image)

The Patofil agent uses the following modalities: hearing, vision, animation, speech, navigation, and mood. These exist as Lisp socket objects, one per modality, that are initialized in the PMLb. These connect the
PMLb to the PMLc and SAL. When the agent’s SAL’ senses something (e.g., when it sees some visible event), a string representation of that percept is communicated to the PMLc, which sends it across the modality-specific socket to the PMLb. Each perceptual modality socket has a separate thread that watches it for new data. When data arrives on the modality socket, it is added to the PMLb percept queue along with an indication of the modality in which it originated. The sense-act loop defined in the agent’s PMLa operates on the percept queue by removing each percept one at a time, converting it into a modality-specific representation for the KL, and initiating a mental act that will result in the agent’s believing the proposition that it saw (or heard, etc) the percept. The complexity of the above description of Patoil’s implementation reflects the complexity of agent-building at present.

5.2 Interactive Tools for Agent Development

Patoil and the other agents in The Trial The Trail are complicated, and their behavior often depends on interactions with each other, with humans (i.e., human participants in the virtual drama), and with other aspects of the environment. The complexity of agent behavior and their dependence on different types of stimuli made it difficult to develop, test, and modify the agents. This problem was exacerbated by the fact that the 3D world in which the agents operate is only available on one or a few computers, requiring agent implementors to be physically present in a particular lab and to perform an involved set-up in order to test agents.

I have developed tools that take advantage of the layered structure of GLAIR to assist with developing and testing agents. One such tool operates by replacing the PMLc and SAL with a Java-based GUI that connects to the PMLb in exactly the same way as the original PMLc does. This GUI allows its user to select from among available percepts and send the messages that correspond to those percepts to the agent via its PMLb. The agent doesn’t know the difference; to the agent’s KL, it appears as if the agent is interacting with the world and actually receiving those percepts. The tool displays a representation of the acts the agent has performed as they reach the “PMLc”, which allows the user to verify that the agent is performing as expected. It also allows timed recording of the sequence of percepts sent to the agent, making it possible to perfectly duplicate sequences of percepts that revealed a bug or other problem.

5.3 Lights/Rats Agents

Another example of the use of MGLAIR agents is the Buffalo Intermedia Performance Studio’s virtual drama performance of Lights/Rats by playwright Suzan Lori-Parks [Parks, 2006]. Lights/Rats is a short play with three agent-actors. The actors have lines to speak, and perform some simple non-verbal acts (e.g., turning to face another actor, moving from one location to another, etc). Each agent actor must know its lines, must be able to perceive when the others have said their lines, and must be able to perceive that another actor has performed certain non-verbal acts.

The Lights/Rats agent-actors derived much of their design and parts of their implementation from the agents in The Trial The Trail, with significant modifications to account for the vastly different script, a change of environment, and the alteration of some modalities. What changed very little between the two projects was the basic structure of the PMLa and PMLb and their connection to the KL. In fact, the implementation for Lights/Rats started with simply copying the PML Lisp files from The Trial The Trail and then changing them to fit with the new performance.

The 2D virtual world the Lights/Rats agents inhabit is implemented in Java using the Karel the Robot software [Bergin et al., 2005], as are their virtual bodies within that world. The agents communicate verbally with each other literally out loud using speech synthesis (with speakers) and speech recognition (with microphones). The speech generation and recognition is implemented using the Microsoft Speech API. In this performance, the world is actually split into two disconnected parts: a Java program in which the agents’ bodies appear as moving on a screen, and the speech/hearing portion of the world, which involves sound waves being generated as a result of agents saying things, moving through air in the real world, and being recognized by the agents.
In this case there is no single version of “the world” with which the agents interact, and the implementation of the SAL is also split into two parts, using different technologies, and running on two different computers. The main difficulty introduced by this configuration is the task of connecting the SAL to the higher layers of the architecture so that the agent’s can perceive and affect the world. This problem was solved by the introduction of a Java-based server that plays the role of PMLc in connecting the SAL to PMLb, as shown in Figure 3.

![Figure 3: MGLAIR in Lights/Rats](image)

5.4 Java Server PMLc

In Lights/Rats, both the PMLb and the SAL components initiate their own connections to the Java PMLc, which listens continually for incoming requests. It has built-in knowledge of the modalities the agents may use, including the directionality of modalities. It also has queues to manage percepts and actions, and separate threads to monitor and update the modality sockets. When percepts originating in the world pass to the PMLc via one of its modality sockets connected to the SAL, the percept is passed to the PMLb on another socket. The PMLc doesn’t care that it has connections from more than one SAL as long as the modality sockets are properly configured and connected.

Advantages of this approach include: the flexibility to easily connect multiple vertically divided layers of the standard GLAIR layers (e.g. two or more SALs), the portability that comes with running one layer (here the PMLc) as a server, to which the other layers simply connect regardless of how many different machines or environments are being used in the different layers; and the ability to reconfigure the layer at runtime without altering code. The Java PMLc described above loads from XML files some of the initialization parameters that determine how it will run, how it handles connections, etc.

However, the current implementation is not as flexible as it could be. In particular, we would like the ability to add and configure modalities either while the program is running or at start-up, neither of which is currently an option. This makes it difficult to reuse the software for different projects, as the Java code
must be modified and recompiled in order to change the configuration.

5.5 Rovio Agent

We have also reused the same basic Lisp-based upper PML \((a+b)\) and their connections to the KL for agents embodied in the real world as physical robots. One such project involved SNePS agents embodied in the WoWee’s Rovio telepresence robot. The Rovio features wheeled locomotion, a webcam, microphone and speaker, infrared object detection, and 802.11g wireless. We connected simple SNePS agent minds to the Rovio hardware using the Java software described above combined with a python API for accessing the hardware’s capabilities. Part of this configuration, absent the KL, was used to control simple scripted Rovio actors in the Intermedia Performance Studio’s production of *WoyUbu* [Anstey et al., 2009].

5.6 Limitations of Existing Agent Implementations

As is discussed more below in section 6.1, the current process of designing and implementing GLAIR agents with some multimodal capabilities requires the agent implementor to write and modify code in a number of different programming languages, and to reimplement functionality that may already exist in other agents at several layers of the architecture.

Worse, there has been no standard approach to the addition, modification, and use of modalities, and to their integration with the reasoning system. There is no built-in notion of modality priority or other mechanism that would allow an agent to attend more closely to some modalities than to others. Though Patofil and the others do use multiple threads to manage modality sockets, the single percept queue within the PML creates a serial bottleneck that prevents the use of a true concurrent model of multimodal perception and action.

6 Proposed MGLAIR Framework

6.1 Moving Forward with MGLAIR

The key distinction between older GLAIR and existing “MGLAIR” agents is the use of multiple modalities within the PML [Shapiro et al., 2005a]. These separate data channels correspond to the sensory and efferent capabilities available to agents (e.g. vision, navigation), which will vary from agent to agent depending on embodiment, environment, and intended tasks.

As described above, these modalities are typically implemented (in Lisp or other languages) as a series of queues and functions that manipulate them within the PML sublayers, and often include message-passing communication among distinct programs that make up the layers. Though this is not strictly part of the architecture, many of our agents recently have used TCP/IP sockets to implement communication between the layers. This approach facilitates the convenient practice of running an agent’s “mind” (the KL) on one computer and connecting to its “body” (the SAL) on another computer via the network. The computer running the SAL may be a workstation or server simulating a virtual world and the virtual bodies that inhabit it, or it may be the on-board computer of a physical robot operating in the real world.

In some cases, we have introduced server software that stands in for PML sublayers (PMLc in particular). This can make it easier to initialize and manage the connections between the layers and has simplified the development of tools that assist with agent creation.

However, the addition of a large and separate piece of software that was created to solve specific problems has made agent design more complicated in some respects. In particular, it has complicated the process of adding, removing and altering modalities.

I aim to replicate this functionality and its advantages as an integrated part of the new MGLAIR framework in a way that eliminates the need to redevelop for each application.

Currently, creating a multimodal GLAIR agent involves copying the source code files of previously-created agents and modifying the relevant functions to fit one’s purposes. Alternatively, one may start from scratch.
and reimplement the layers and the mechanisms that handle the multiple modality registers. If the PML sublayers are to communicate via sockets, one must implement functions to initialize and manage socket communication either directly between layers, or by connecting both sides to some intermediary server software. In either case, the task may involve network programming, inter-process communication, or some other approach to directly managing communication between the layers. Ideally, MGLAIR agent designers would be able to largely ignore these low-level implementation details and focus instead on the tasks of agent design and implementation.

Therefore, a major goal for this project is the development and implementation of a standard framework for MGLAIR that will automate network communication setup and other low-level details during agent creation using reasonable default behaviors. These defaults will be user-modifiable without requiring the user to program in Lisp, Java, etc. in order to effect simple deviations from the defaults.

6.2 MGLAIR Framework Features

6.2.1 Multimodal Capabilities

Explicit support for the concurrent use of multiple modalities is a key goal of this proposal.

Sensing and perceiving in a modality necessarily involves the flow of information up from the agent’s SAL, through the PML, and into the KL, where it becomes available to conscious reasoning. Along the way, the SAL’s raw sense data must be transformed. Sense data is assembled into perceptual structures in the PML, which then become symbolic structures suitable for use by the KL.

Likewise, acting in a modality involves the initiation of an act by the KL that causes the flow of high-level commands down through the layers of the architecture until they are transformed into, e.g., simple motor commands that are executed by the SAL.

This communication across the layers of the architecture is divided into separate data channels, each corresponding to a distinct modality — one for each perceptual or efferent capability of the agent. The modalities available to an agent are limited by its embodiment. Each modality’s data channel is independent of the others. Each modality has its own properties that determine how data is handled within the modality and, to some extent, how data from that modality is integrated with knowledge and planning. These may include specifications of how much data a single modality can process at one time and how to resolve conflicts when an efferent modality in the process of carrying out one act is interrupted by another act command. Also among these modality properties are direction, priority, and structured information about the type and formatting of data it conveys.

This treatment of modalities, combined with mechanisms to integrate their contents with the reasoning and acting systems in a way that allows truly concurrent access, will simplify the creation of agents capable of sensing in multiple modalities at once. It will also facilitate the use of true multimodal perception in which percepts from different, but related, modalities can be combined to form a more realistic model of the world than would otherwise be available.

Acting will also be improved and simplified by this approach. Currently, issues surrounding multimodal action and the interactions between different types of acts are handled by the agent designer on a case-by-case basis. For instance, the Fevahr Cassie GLAIR agent described in [Shapiro, 2003] makes the distinction between durative and punctual acts, and provides a mechanism in the PML for temporarily interrupting a durative act to perform a punctual act. Without separate efferent modalities and built-in mechanisms for managing their interactions, it is difficult to handle this sort of act interaction in a general way. Rather, the implementation is hard-coded in the primitive action functions, and the function for each act must specifically address the question of its interactions with each other act the agent may perform. Managing this sort of interaction between acts will be greatly simplified by the architecture here proposed.

6.2.2 Towards an Implementation of Modality

Though some existing GLAIR agents use multiple modalities, the implementation of this feature is a combination of sockets, functions, threads, and queues in Lisp and even Java source files. These lack a unifying
structure and are of limited reusability and scalability.

The solution to this is a conceptualization (and eventual implementation) of *modality* as a type of object that agent designers can instantiate when assembling agents, along with mechanisms in the PML for manipulating such objects.

The proposed project will coincide with the ongoing development of SNePS 3, the next major version of SNePS. SNePS and, in particular, its acting system are integral parts of MGLAIR. As discussed more in section 6.3, part of the current proposal is the implementation of the SNePS 3 acting system. The implementation of SNePS 3 departs from SNePS 2 in that it makes full use of Lisp’s CLOS (Common Lisp Object System) objects where appropriate. I plan to organize modalities as CLOS classes that subclass a general modality class. The implementation will include standard modalities typically used by MGLAIR agents (e.g., vision and speech) with default settings for those modalities’ basic parameters (direction, type of message data, etc).

This modularity will simplify the selection, creation, addition, configuration, and removal of modalities during agent implementation and execution. It will also simplify the management within the framework itself of separate modalities and the integration of percepts and acts with reasoning. Ideally, agent design and implementation should be so simple that even a computational agent could do it [Goldfain et al., 2006]. One can even imagine an agent coming to realize through acting and reflection that it should add a modality to its repertoire, and then adding the modality itself.

Since it has been useful in the past to run different layers of the architecture on different physical machines, MGLAIR must support this functionality. The simplified view of modalities as single CLOS classes outlined above will not directly support this decoupling of layers. Recall that in recent GLAIR agents, a modality within the PML has consisted of Lisp code in the upper layers and Ygdrasil, Java, or some other software playing the role of PMLc and communicating with the upper layers and SAL via TCP/IP sockets.

### 6.2.3 Standardized Layer Interactions

One way to achieve this decoupling of the PML sublayers while retaining the modularity of modalities and the notion of modality as a single whole is to create one (or more) standard implementation of a network-accessible communications layer (in Python, Java, or even Lisp). This implementation would need a similar notion of modalities as objects, though the tasks performed by these objects would be different from those in the PMLa and PMLb, and would involve communicating with them. It would also need a way of interacting, (preferably at run-time or later) with the upper layers in order to agree on the modalities in use and their features (especially directionality and the type of data structures to be transmitted). This could take the form of an exchange of XML configuration files or other structured information.

Because it will be useful to allow agents to run in a number of different configurations and environments, it may make sense to have multiple implementations of this network-accessible layer. However, if they are to be integrated with MGLAIR as a framework, and the SAL and PML sublayers are to communicate seamlessly with them, the configuration and behavior of the implementations must be constrained by a standard. Therefore, the first step in the design of this portion of the framework must be the development of a PML standard that determines the format and features of modalities and the nature of communications between the layers. This will specify, among other things, the general behavior of modality objects at each level and the mechanism by which instantiated modalities may be shared between layers. This standard and implementation(s) will provide agent developers with the tools to rapidly develop multimodal agents often without writing or modifying any code in the upper PML. It will also leave open the possibility of a developer creating his own PMLc that, so long as it adheres to the standard, will interoperate with the other layers without modifying them at all. The SAL is inherently bound to the environment and agent’s embodiment and will often involve completely different implementations for any two agents. The PML standardization will constrain the SAL only with respect to its interactions with the PML.
6.2.4 Layer Interactions

MGLAIR agents recently have used TCP/IP sockets to achieve communication between layers of the architecture. One benefit of this approach is the flexibility and portability it provides. The MGLAIR framework will preserve this flexibility without requiring agent designers to also be network programmers, as discussed above. In addition to specifying how these connections are managed, the standardization of these communication channels will allow, and perhaps require, the structured exchange of perceptual and efferent data. Raw TCP/IP sockets may not be ideal for exchanging this type of information. One possibility is the use of XML-based SOAP (Simple Object Access Protocol) [Gudgin et al., 2001] messages.

6.2.5 Prioritizing Percepts and Acts

One modality feature that will be immediately useful is priority. Depending on the environment, the current context or situation, what an agent intends to do, and on its current task, it will often be the case that an agent may want to pay more attention to some modalities than to others. These modality priorities may be defined as immutable for a particular kind of modality or particular instance of that type, or they may be alterable by the agent designer at the time of modality selection, or even by the agent during the course of reasoning and acting.

While not itself a model of attention, a mechanism that allows agents to selectively attend to modalities is a necessary precursor to the development of such a model. Implementations of attention, whether at the conscious or subconscious levels (or some combination), will require the ability to filter out and ignore those percepts that are not the focus of attention. The development and implementation of a model of attention for MGLAIR agents is outside the scope of the current proposal.

6.2.6 Modality Management

Humans and other animals are able to simultaneously process and integrate perceptual information from different sources/modalities, and to act in multiple modalities. Support for multiple concurrently-processed perceptual and efferent modalities will expand SNePS agents’ capabilities to include types of thinking and behavior that were previously difficult or impossible to implement.

The task of managing these modalities within the architecture and integrating their management with reasoning and acting will require significant research and development effort. The implementation will likely include percept and act queues with dedicated threads to attend to each modality, to propagate acts downward and percepts upward to/from the lower layers. It will also include functions to initialize modalities and associate them with acts in the KL. I propose to build into the MGLAIR framework a sense-act loop that includes the notion of priority as a property of modalities that affects the way in which acts are executed in the world and percepts are perceived in the mind.

6.2.7 Modularity

Some architectures place more importance than others on a particular kind of modularity. Descriptions of ACT-R, for instance, emphasize the fact that it is comprised of several interconnected modules, each of which is supposed to correspond to a particular region of the human brain [Anderson et al., 2004]. Though the structure of brains has not directly influenced the number or natures of subsystems within (M)GLAIR, the SNePS knowledge layer alone is comprised of multiple distinct modules that conspire to produce reasoning. Modularity is also seen in MGLAIR in the separation of the architecture into independent layers, some of which may completely ignore the implementation details of the other layers. The division of modalities into distinct objects each having its own configurable properties and its own separate channel for percepts/acts also contributes significantly to the overall modularity of the system.

This development of modalities as standard parts of the system should in some cases facilitate the development of separate modules to perform specified types of reasoning. For instance, some architectures include among their modules one dedicated to visual processing. Though vision is a key capability for embodied agents operating in the real world or in many virtual/simulated worlds, I do not plan to include...
a “visual processing module” as a part of the MGLAIR architecture. Rather, the system supports flexible modalities that might be used as hooks into such subsystems.

One can imagine a visual processing system designed for use with MGLAIR that uses traditional (i.e., statistical) image processing and computer vision techniques. Connecting such a system to an MGLAIR agent via bidirectional vision modalities might allow the agent’s conscious reasoning system to both perceive and directly or indirectly influence perception of the visual world. A similar approach might be used to achieve integration with connectionist-based subsystems and a sort of de facto hybridism for a number of different applications (language processing, some forms of learning, etc).

The design, implementation, and integration with MGLAIR of any such subsystem would be non-trivial and is not part of the current proposal. However, since MGLAIR is an architecture for embodied agents, it must leave open support for the addition over time of increasingly realistic perceptual modules and ways of interacting with the world. This requirement will be a major consideration during the design and implementation of the system. One way this will be achieved is by leaving open to agent designers the possibility of implementing their own PMLs and SALs, provided that the layer interactions are compatible with layer interaction standards (c.f. sections 6.2.3, 6.2.4). These standards will not constrain the types or formats of data passed between SAL and PML layers and sublayers within modalities, but will only require that each non-standard modality has a structured description that can be understood by all the layers.

6.3 SNePS 3 Acting System

SNeRE (the SNePS Rational Engine) is the SNePS subsystem that handles planning and acting [Kumar, 1993] [Shapiro and The SNePS Implementation Group, 2007], and it is a key part of GLAIR. It expands SNePS agents’ capabilities by connecting logic-based reasoning with acting. This makes it possible for agents to decide in the course of reasoning to perform some action and then to actually perform it. The plans formed and actions taken by an agent at any time depend in part on the agent’s beliefs (including beliefs about the world based on its perceptions) at that time.

SNeRE also provides the basis for the Knowledge Layer’s connection to the lower layers of GLAIR. Without SNeRE, all SNePS agents would be essentially disembodied ‘minds’ disconnected from the world. Actions supported by SNeRE include mental acts (believing or disbelieving a proposition; adopting or unadopting a policy), control acts (control structures such as achieve), and external acts such as moving, sensing, or otherwise interacting with the external world.

SNePS 3 is the latest version of SNePS, currently under development by Shapiro and the SNePS Implementation Group. It is based on Shapiro’s Logic of Arbitrary and Indefinite Objects [Shapiro, 2004], which has syntax and semantics different from those used in previous versions of SNePS.

The acting system for SNePS 3 has yet to be implemented. As the acting system has an integral role in GLAIR and, by extension, in MGLAIR, the design and implementation of a new version of SNeRE for SNePS 3 will be the first major task in our proposed development of MGLAIR.

6.3.1 Policies and Propositions

While the specifics of SNeRE 3 are still to be determined, and it will be largely derived from SNeRE 2, one important issue that must be addressed in this new implementation is the distinction between propositions and policies. Historically, SNePS has not made a strong enough distinction between these two types of mental entities.

SNePS policies are condition/action (or proposition/action) pairs similar to productions in traditional production systems. For instance, the SNePS policy \texttt{whendo(p, a)}, when held by an agent, will cause the agent to take action \texttt{a} and then remove that policy from its knowledge base if it comes to believe the proposition \texttt{p} as a result of forward inference in the course of reasoning. A \texttt{wheneverdo(p, a)} policy is similar but is retained after it has been used [Shapiro and The SNePS Implementation Group, 2007].

The above discussion mentions an agent “holding” or “removing” policies. It would be less correct to say that an agent might believe or disbelieve a policy, or that a policy might be asserted in an agent’s knowledge base. However, policies have often been treated too much like propositions in SNePS 2. For instance, the
SNePS 2.7 manual defines \texttt{whendo} as: \texttt{whendo(p, a)}: If SNIP forward chains into \texttt{p}, perform \texttt{a}, and then disbelieve the \texttt{whendo} [Shapiro and The SNePS Implementation Group, 2007].

This shows that the current version of SNePS does not make a strong distinction between propositions and policies. However, we have come to regard them as two different kinds of entities [Shapiro and Bona, 2009]. Since policies, unlike propositions, may not denote truth values, they are distinct and should be represented differently in the system.

For example, the following is a valid formula in SNePS 2.7 that might be used to express something like, “For any \(x\), if \(x\) is a person, then I hold the policy that when I am near that person, I greet that person”:

\[
\text{all}(x) \left( \text{Person}(x) \Rightarrow \text{whendo}(\text{Near}(I, x), \text{Greet}(x)) \right).
\]

The conditional is a logical connective. Any ground instance of this conditional formula, if it is to denote a proposition, should have proposition-denoting terms as its constituents. No \texttt{whendo} or other policy-denoting term ever denotes a proposition. Yet this formula appears to be a universally quantified conditional with a \texttt{whendo} policy as its consequent. When this is allowed, the semantics become unclear, and it is difficult to say precisely what the terms represent.

Therefore, one major goal for the implementation of SNeRE 3 will be a treatment of policies that distinguishes them from propositions while preserving their connection to the reasoning system. Though it should not be possible for agents to \textit{assert} or \textit{believe} policies in the same way they would a proposition, the system must retain agents’ ability to \textit{adopt} or \textit{unadopt} policies in the course of reasoning about beliefs.

The exact implementation and other specific details of how this will be addressed in SNeRE 3 remain open to further investigation. Preliminary work on this suggests a solution inspired by the \texttt{withall} control act, which is currently a part of SNeRE. \texttt{withall}(x, \texttt{p}(x), \texttt{a}(x), \texttt{da}) combines a variable \(x\), an open proposition \(p(x)\), an act to be performed on entities of which the proposition is true \(a(x)\), and a default act \(da\) [Shapiro and The SNePS Implementation Group, 2007]. The new approach to policies and their integration with propositions will likely take a similar form, including variable binding within the policy itself. That is, \texttt{whendo} policies in SNeRE 3 will take a form similar to the following:

\[
\text{whendo}(x, \text{Person}(x) \text{ and } \text{Near}(I, x), \text{Greet}(x)).
\]

6.4 Agent Wizards and Templates

This organization of the MGLAIR architecture into a standard framework will allow agent designers to create new MGLAIR agents without manipulating code at the level of the KL or upper layers of the PML. One way this might be achieved is through the use of agent templates: basic agents with common modalities in place and a stripped-down KL. Another option is the use of agent wizards: either graphical or simple text-based user interfaces that step the user through a sequence of design choices (selection of basic primitive acts, selection of associated modalities and their features, etc) and then generates a skeleton PML based on the user’s selections. If the environment is not one previously used, then the SAL will need to be implemented and integrated with the PMLc. However, this wizard might be packaged with a standard world simulation or robot hardware with established SAL implementations. This will be ideal for instructional purposes for tutorials or courses in which agent creation is the main goal. It will also naturally facilitate the rapid development and testing of agents to perform useful cognitive tasks as part of some real-world project, and of agents created to test theories about cognition.

7 Summary

I have proposed to develop both (1) a full theory and specification, and (2) an implementation in software of MGLAIR - a multimodal, grounded, layered, architecture with integrated reasoning. This architecture will serve as a general-purpose platform for the implementation of embodied cognitive agents. A major subtask of this project is (3) the implementation of SNeRE3, the SNePS 3 acting system.

It is difficult to provide clear metrics this far in advance that will allow me to determine when the proposed project is completely finished. It will not be finished before I have developed a theory and specification of
the architecture that include the features described in this proposal, and a fully functional implementation of the architecture. This implementation will then be tested by developing agents in multiple domains that use various combinations of modalities to reason about, and perceive and act within, their respective environments. The success of the architecture will be judged based on the ease of agent creation; its flexibility and adaptability at supporting agents with diverse sets of modalities and reasoning capabilities in at least a few distinct domains and environments; and the complexity of the cognitive behaviors and capabilities these agents possess. The last of these criteria is necessarily vague, but the agents ought at least to be better than their pre-MGLAIR predecessors at performing similar tasks. This may be tested by re-implementing some agents using the new architectural framework and comparing the results to the old versions.

8 Proposed Timeline

- *Summer, Fall 2010* - Complete full cognitive architectures literature review; more fully investigate notion of modality in cognitive psychology and other relevant fields.
- *Fall 2010* - Design and implement SNeRE3; begin theoretical work and preliminary design of MGLAIR framework.
- *Spring 2011* - Conclude design and begin software implementation.
- *Summer, Fall 2011* - Conclude implementation; test framework by creating agents in multiple domains.
- *Spring 2012* - Conclude testing; write results and other final sections of dissertation; edit.
References


